Dheeraj Yenamala

Capstone Technical Report

Acquiring the Data

The topic of my capstone project is the Higgs Boson Machine Learning Challenge presented on Kaggle. As such, the data needed for the project was available in both a training data set and test data set. I loaded the training data csv file into Jupyter Notebook as a pandas dataframe. The training data set for the Kaggle competition contains 250,000 events with an ID column, thirty feature columns as well as a weight column and the label column. The Higgs Boson Machine Learning Challenge is a classification task in which competitors are challenged to use the available feature columns to build a classification model to determine whether an event is either signal or background noise.

Preprocessing and Cleaning

As the 'Label' column contained binary values in the form of either 's' for 'signal' or 'b' for 'background noise', I converted the 's' values to 1 and the 'b' values to 0 using a lambda function. Kaggle competition data sets typically are in good shape and have been cleaned, and in this case that notion was for the most part true. The data set did contain null values, however, in the form of values set to equal -999. The Kaggle competition page stated that values that were unusable or could not be computed were set to -999, a value that was outside the normal range for all values. I converted the values set to -999 to 'NaN' using the .replace method.

There were a significant amount of null values in several of the feature columns. Some of the feature columns contained 177,457 values, while others contained 99,913 values. Since the each columns contained 250,000 values, I thought that I could use the feature columns which contained 99,913 null values as long as I could fill in the null values. I used both the Imputer or Interpolate techniques to insert number values into the null values of the feature columns. For the Imputer technique, I created two dataframes, in which one of them I imputed the median value of the column to fill null values, and in the other dataframe I imputed the most frequent value of the column to replace null values. Although this technique of using the Imputer can fill null values to make the feature column usable, it is not very robust. However, the Interpolate technique would use the other values of the feature columns to estimate possible values for the null values, and this method is much more robust and has led to better results.

I selected the PRI feature columns with no null values, as well as the PRI feature columns which contained 99,913 null values which were interpolated as the X variable to feed into a model. The y variable was set equal to the binarized 'Label' column. I then used train_test_split to split up the data set into training and testing data. Standard Scaler was also used to scale fit and transform the X_train data and transform the X_test data.

Modeling Process

Before setting up the classification models, I established a baseline accuracy to set a benchmark of accuracy that I needed to improve upon with my models. The baseline accuracy was approximately 65.73% meaning that were was the influence of a moderate class imbalance. There were several classification models that I tested and observed to determine which model could produce the best accuracy in terms of classification accuracy. I used the Gradient Boosting Classifier, Random Forest Classifier, Linear Support Vector Machines Classifier, Logistic Regression as well as using CalibratedClassifierCV in conjunction the models, specifically Linear Support Vector Machines Classifier as the LinearSVM Classifier does not have a predict.proba_ method to construct a ROC curve and calculate AUC scores. I also used GridSearchCV in conjunction with a select few of the models to determine the best hyperparameters with which to tune the model.

Evaluation

In terms of accuracy scores, the Random Forest Classifier performed the best, with accuracy scores in the range of 0.75 to 0.79. I also opted to use a ROC curve plot to determine the accuracy of my model as a class imbalance can have en effect on how accurate an accuracy score depicts the performance of a model. The ROC curve depicted a fairly accurate model, as the AUC, or are under the curve, score was .79, and the ROC curve showed that model was a good deal more accurate than a 0.50 chance baseline prediction by chance. Although the results of the model showed that it was quite accurate, the confusion matrix of the model showed that the model was misclassifying a large number of events of a presence of signal as background noise.

Next Steps

Due to the presence of many misclassifications of signal as background noise, I need to try to incorporate a wider range of feature engineering methods to ensure a high rate of accuracy. I also have implemented Principal Component Analysis into the preprocessing of my data, and intend to incorporate the results of the analysis to enhance the feature selection portion of the project.